Published in: Butt, Miriam/King, Holloway (eds.): Proceedings of the 14th International Lexical Functional Grammar Conference (LFG 2009). Cambridge, United Kingdom. July 13 - 16, 2009. - Stanford: CSLI Publications, 2009. 21 pp.

AUTOMATIC ACQUISITION OF LFG RESOURCES FOR GERMAN - AS GOOD AS IT GETS

Ines Rehbein and Josef van Genabith
Universität des Saarlandes Dublin City University

Abstract

We present data-driven methods for the acquisition of LFG resources from two German treebanks. We discuss problems specif c to semi-free word order languages as well as problems arising from the data structures determined by the design of the different treebanks. We compare two ways of encoding semi-free word order, as done in the two German treebanks, and argue that the design of the TiGer treebank is more adequate for the acquisition of LFG resources. Furthermore, we describe an architecture for LFG grammar acquisition for German, based on the two German treebanks, and compare our results with a hand-crafted German LFG grammar.

1 Introduction

Traditionally, deep, wide-coverage linguistic resources are hand-crafted and their creation is time-consuming and costly. Much effort has been made to overcome this problem by automatically inducing linguistic resources like rich, deep grammars, lexicons and subcategorisation frames from corpora. Most work so far has concentrated on English, like that of Hockenmaier and Steedman [2002], Nakanishi et al. [2004] and Cahill et al. [2002, 2004]. They present successful approaches for the acquisition of deep linguistic resources from the Penn-II treebank, using different grammar frameworks like CCG, HPSG and LFG. English, however, is a conf gurational language, where strict word-order constraints help to disambiguate predicate-argument structure. Porting these approaches to a semi-free word order language, we have to ask: How good can it get? Can we expect similar results when dealing with (semi-) free word order? Can data-driven methods cope when dealing with ambiguous data structures and sparse data, caused by a rich(er) morphology in combination with case syncretism? And, furthermore, what impact does treebank design have on the automatic acquisition of linguistic resources like deep grammars?

This paper describes approaches to treebank-based acquisition of LFG resources for a semi-free word order language, based on the method of Cahill et al. [2002, 2004, 2008], Burke et al. [2004] and O'Donovan et al. [2005], who presented the large-scale acquisition of LFG grammars and lexical resources from the English Penn-II and Penn-III treebanks. They also presented work on data-driven multilingual unif cation grammar development for Spanish, Chinese and German. While results point to treebank-based grammar acquisition being a universal method, results for other languages are by far lower than the ones achieved for English and the English Penn treebank.

There are different possible reasons for this: f rst of all, the size of the English Penn-II treebank, which is much larger than most treebanks for other languages, might be responsible for the good results on English. Another reason might be the conf gurational English word order, where strict constraints determine the grammatical function of a lexical unit in a certain surface position. Finally, the good results for English might be due to the data structures employed in the Penn-II

treebank, which might be optimised for the task at hand and thus improve performance on the English data.

In this paper we develop different f-structure Annotation Algorithms for German, based on two German treebanks with crucially different annotation schemes, adapted to feature sets of varying granularity as represented in three different gold standards. We discuss problems specif c to the annotation schemes of the two treebanks as well as to language-specif c properties of German, where the variability in word order and the richer morphology (compared to English) often result in data sparseness, causing severe problems for data-driven methods. Finally, we compare the performance of our data-driven grammar acquisition architectures with the hand-crafted German ParGram LFG of Dipper [2003], Rohrer and Forst [2006], and Forst [2007].

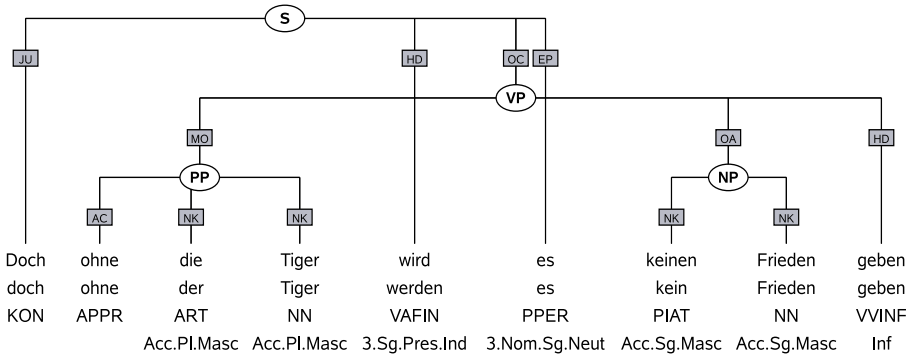
The paper is structured as follows: Section 2 gives an overview of typological properties of German and their representation in two different German treebanks. Section 3 describes the LFG grammar acquisition architecture for German, focusing on the differences to the work of Cahill et al. [2003, 2005] and Cahill [2004]. Section 4 reports on the automatic generation of LFG f-structures and discusses problems specif c to semi-free word order and to the design of the German treebanks. Section 5 presents a comparison of our best automatically acquired LFG grammar with related work, namely the hand-crafted ParGram LFG for German. The last section concludes.

2 Typological Properties of German and their Representation in Two German Treebanks

German, like English, belongs to the Germanic language family. Despite being closely related, there are crucial differences between the two languages. One of them is the semi-free word order in German, which contrasts with the more conf gurational English; another, but related difference concerns the richer morphology in German, compared to the rather impoverished English morphology. Both properties are ref ected in the treebank data structures used to represent syntactic analyses of the particular languages.

2.1 TiGer and TüBa-D/Z: Two German Treebanks

The TiGer treebank [Brants et al., 2002] and the TüBa-D/Z [Telljohann et al., 2005] are two German treebanks with text from the same domain, namely newspaper text. Both treebanks are annotated with phrase structure trees, dependency (grammatical relation) information and POS tags, using the Stuttgart Tübingen Tag Set (STTS) [Schiller et al., 1995]. Differences regard the set of categorial node labels used for syntactic annotation and the set of grammatical function labels. TiGer annotates 25 different syntactic categories and distinguishes between 44 different grammatical functions, while the TüBa-D/Z uses 26 different syntactic categories and 40



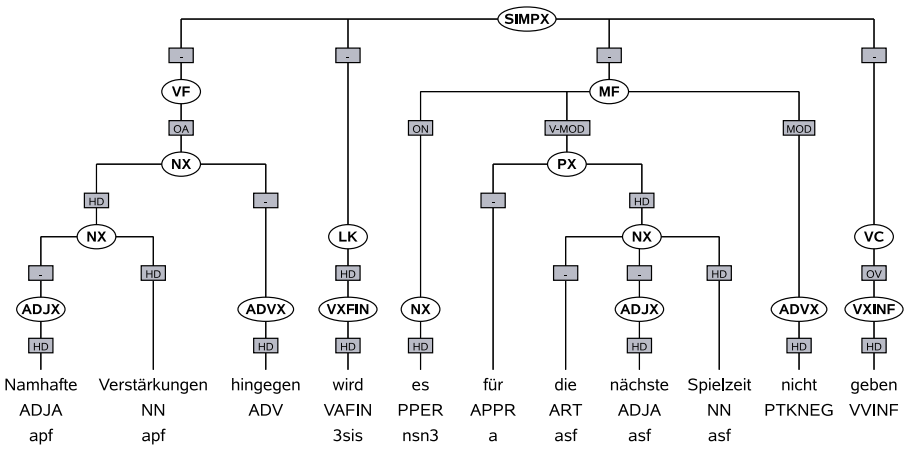
"But without the Tigers there will be no peace"

Figure 1: TiGer treebank tree

grammatical function labels. The main differences between the two treebanks are: (1) the flatter annotation in TiGer compared to the more hierarchical annotation in TüBa-D/Z, (2) the annotation of unary nodes in the TüBa-D/Z and no unary nodes in TiGer, (3) TüBa-D/Z uses topological fields to annotate the semi-free German word order, which allows for three possible sentence configurations (verb-first, verb-second and verb-final), and (4) TiGer annotates Long Distance Dependencies through crossing branches, while TüBa-D/Z encodes LDDs with the help of grammatical function labels (see Figures 1 and 2).

3 Automatic Annotation of LFG F-Structures

Cahill et al. [2003, 2004, 2005, 2008] presented a modular architecture for automatically annotating the English Penn-II treebank with LFG f-structures (Figure 3), which enables them to automatically extract deep, wide-coverage grammars which yield results in the same range as the best hand-crafted grammars for English [Briscoe and Carroll, 2002, Kaplan et al., 2004]. The f-structure Annotation Algorithm (AA) exploits lexical head information, and categorial, configurational and functional information as well as traces and co-indexation annotated in the Penn-II treebank. After determining the head of each constituent, the main module of the AA uses *left-right context annotation principles* to assign the most probable f-structure equation to each node in the tree (Figure 3). These principles express annotation generalisations and have been hand-crafted by looking at the most frequent grammar rules for each node in the Penn-II treebank and are also applied to unseen low-frequency rules. A sample partial left-right context annotation rule for NPs is given in Table 1. The left-context rule states that all adjectives or adjectival phrases to the left of the head of an NP should be annotated as an adjunct, while the right-context rule specifies that an NP to the right of the head of an NP is an



"However, there won't be considerable reinforcements for the next playing season."

Figure 2: TüBa-D/Z treebank tree

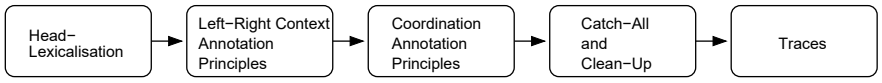


Figure 3: Architecture of the English f-structure Annotation Algorithm (AA)

apposition. The creation of these left-right-context rules needs linguistic expertise and crucially depends on conf gurational properties of English.

left-context	head	right-context
JJ, ADJP: $\downarrow = \in \uparrow$ ADJUNCT	NN, NNS, ... $\uparrow=\downarrow$	NP: $\downarrow = \in \uparrow$ APP

Table 1: Left-right context annotation rule used in the English AA

Coordinations are treated seperately. After adding f-structure equations to all nodes in the tree, the *Catch-All and Clean-Up* module deals with overgeneralisations. Finally, traces are resolved.

The German LFG AA, like the English one, is highly modularised and proceeds as follows (Figure 4). First it reads in the treebank trees encoded in the NEGRA export format and converts each tree into a tree object. Then it applies head-f nding rules which we developed in the style of Magerman [1995], in order to determine the head of each local node.[1] The head-f nding rules specify a set of candidate heads, depending on the syntactic category of the node, and also the

[1] TiGer provides head annotation for all categorial nodes except NPs, PPs and PNs. Due to the f at annotation in TiGer, partly resulting from the decision not to annotate unary nodes, the problem of identifying the correct head for those nodes is more severe than for the TüBa-D/Z, where the more hierarchical structure results in smaller constituents which, in addition, are all head-marked. When annotating original treebank trees, the head-f nding rules are applied to NP, PP and PN nodes; when

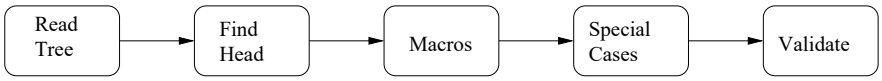


Figure 4: Architecture of the German f-structure Annotation Algorithm

direction (left/right) in which the search should proceed. For prepositional phrases, for example, we start from the left and look at all child nodes of the PP. If the left-most child node of the PP has the label KOKOM (comparative particle), we assign it as the head of the PP. If not, we check if it is a preposition (APPR), a preposition merged with a determiner (APPRART), an apposition (APPO), and so on. If the left-most child node does not carry one of the candidate labels, we take a look at the next child node, working our way from left to right.

For some of the nodes these head-finding rules work quite well, while for others we have to accept a certain amount of noise. This is especially true for the flat NPs in the TiGer treebank. A *Special Cases* module checks these nodes at a later stage in the annotation process and corrects possible errors made in the annotation.

After determining the heads, the tree is handed over to the *Macros* module which assigns f-structure equations to each node. This is done with the help of macros. Sometimes these macros overgeneralise and assign an incorrect grammatical function. In order to deal with this, the *Special Cases* module corrects inappropriate annotations made by the *Macros* module. Finally the *Validation* module takes a final look at the annotated trees and makes sure that every node has been assigned a head and that there is no node with two child nodes carrying the same governable grammatical function.

The most important difference in the design of the English and the German AAs concerns the application of left-right context annotation rules described above. For English, these rules successfully specify the correct annotation for the majority of local nodes in a given tree. For German, however, these rules do not work as well as for English. Table 2 illustrates this point by showing different possibilities for the surface realisation of a (rather short) German sentence. Some of the examples are highly marked, but all of them are possible surface realisations of (1).

(1) Die Anklage legt ihm deshalb Betrug zur Last.
the prosecution lies him therefore fraud to the burden.
The prosecution therefore charges him with fraud.

The f-structure-annotated grammar rule for the sentence in (1) (Figure 5) tells us that the first NP *Die Anklage* (the prosecution) is the subject of the sentence,

running the AA on parser output trees with erroneous or no GF labels in the trees, we also make use of head-finding rules for other syntactic categories.

In TüBa-D/Z, heads are marked for most categorial nodes. However, there are some open issues, like the one concerning the head of the middle field or of proper name nodes, or the annotation of appositions, which are considered to be referentially identical and therefore bear no head marking in the TüBa-D/Z.

S	→	NP	VVFIN	PPER	PROAV	NN	PP
		↑ SUBJ=↓	↑=↓	↑ DA=↓	↓∈↑ MO	↑ OA=↓	↑ OP=↓

Figure 5: Grammar rule and f-structure equations for the sentence in (1)

Die Anklage	legt	ihm	deshalb	Betrug	zur Last.
Betrug	legt	ihm	deshalb	die Anklage	zur Last.
Ihm	zur Last	legt	die Anklage	deshalb	Betrug.
Zur Last	legt	ihm	die Anklage	deshalb	Betrug.
Deshalb	legt	ihm	die Anklage	Betrug	zur Last.
...	...	...	...	...	...

Table 2: Variable word order in German (sentence (1))

while the noun *Betrug* (fraud) should be annotated as an accusative object, and the pronominal adverb *deshalb* (therefore) is an element of the modif er set. Table 2, however, illustrates that these constituents can occur in very different positions to the left or right of the head of the sentence. This shows that, unlike for a strongly conf gurational language such as English, the specif cation of left-right-context rules for German is not very helpful.

Instead of developing horizontal and strongly conf gurational context rules, the AA for German makes extended use of macros, using different combinations of information such as part-of-speech (POS) tags, node labels, edge labels and parent node labels (as encoded in the TiGer and TüBa-D/Z treebanks). First we apply more general macros assigning functional annotations to each POS, syntactic category or edge label in the tree. More specif c macros, such as the combination of a POS tag with the syntactic node label of the parent node or a categorial node with a specif c grammatical function label, can overwrite these general macros. The order of these macros is crucial, dealing with more and more specif c information. Some of the macros overwrite information assigned before, while others only add more information to the functional annotation.

To give an example, consider the POS tag ART (determiner). The f rst macro is triggered by this POS tag and assigns the f-structure equation $\uparrow{=}\downarrow, \downarrow \mathit{det\text{-}type} = \mathit{def}$. The next macro looks at combinations of POS tags and grammatical function (GF) labels and, for a determiner with the label NK (noun kernel), adds the equation $\uparrow \mathit{spec} : \mathit{det} = \downarrow$, while the same POS tag gets assigned the functional equation $\downarrow \in \uparrow \mathit{spec} : \mathit{number}$ when occurring with the edge label NMC (numerical component). The annotation for the combination of POS and grammatical function label can be overwritten when a more specif c macro applies, e.g. one which also considers the parent node for a particular POS-GF-combination.

The determiner with edge label NK has so far been annotated with $\mathit{headword}, \downarrow \mathit{det\text{-}type} = \mathit{def}, \uparrow \mathit{spec} : \mathit{det} = \downarrow$. This is overwritten with the f-structure equation $\uparrow \mathit{obj} : \mathit{spec} : \mathit{det} = \downarrow$, if it is the child of a PP node. This is due to the fact that the annotation guidelines of the TiGer treebank analyse prepositions as the head of a PP, while the head noun (and its dependents) inside the PP is annotated as the

object of the preposition. Due to the flat annotation in the TiGer treebank, it is not helpful to use vertical context above the parent node level. The AA makes heavy use of the *Special Cases* module, where further annotation rules are specified for most syntactic categories. One tricky case is that of NPs, which have a totally flat structure in the TiGer treebank. There are many cases where the information about POS tag and grammatical function label is not sufficient, and neither is their relative position to the head of the phrase. In those cases the presence or absence of other nodes decides the grammatical function of the node in question.

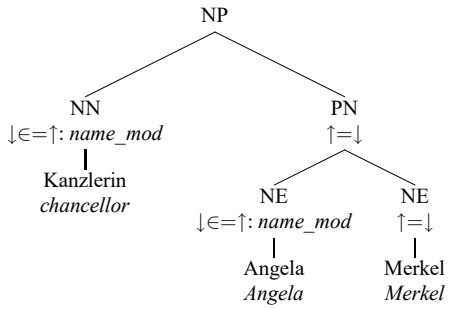


Figure 6: NP-internal structure in TiGer (PN=head)

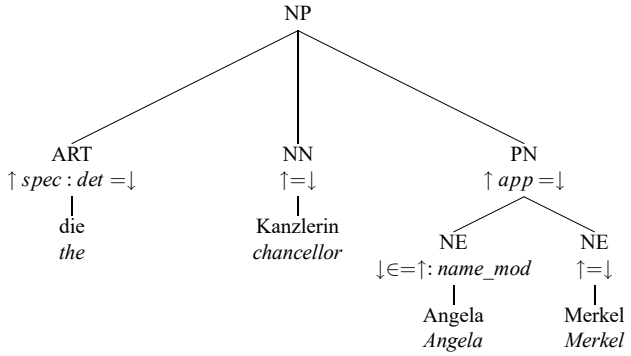


Figure 7: NP-internal structure in TiGer (PN=apposition)

To illustrate this, consider the three examples in Figures 6-8. All three examples show an NP with a noun child node followed by a proper name (PN) node, but where the grammatical annotations differ crucially. In Figure 6, the PN is the head of the NP. In Figure 7, where we have a determiner to the left of the noun (NN), the noun itself is the head of the NP, while the PN is an apposition. The third example (Figure 8) looks pretty much like the second one, with the exception that *Merkel* is in the genitive case. Here the PN should be annotated as a genitive attribute. This is not so much a problem for the annotation of the original treebank trees where we have both the correct grammatical function labels as well as morphological information. For parser output, however, morphological information is not available and the grammatical functions assigned are often incorrect. In Section 4.2.1

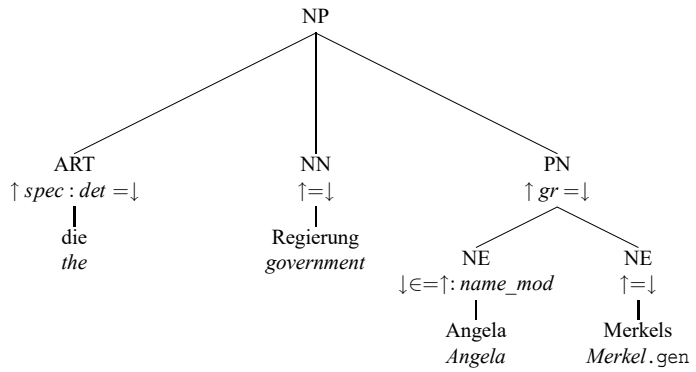


Figure 8: NP-internal structure in TiGer (PN=genitive to the right)

we will return to this issue und discuss the reason for the missing morphological information in the parser output.

3.1 Differences between our AA for German and Preliminary Work

The annotation algorithm for German presented in this chapter is based on and substantially revises and extends preliminary work by Cahill et al. [2003, 2005] and Cahill [2004]. The AA by Cahill et al. provides annotations for a rather limited set of grammatical functions only (26 grammatical functions: 11 governable functions, 10 non-governable functions and 5 atomic features). We created a new gold standard f-structure bank containing 250 sentences from the TiGer treebank, the TIGER250, which uses a substantially extended set of grammatical functions and features (46 grammatical functions: 14 governable grammatical functions, 13 non-governable grammatical functions and 19 atomic features). As a result, the annotated resources contain richer linguistic information and are of higher quality and usefulness compared to the one of Cahill et al. [2003, 2005] and Cahill [2004]. Our annotation algorithm also makes use of a valency dictionary in order to distinguish between stative passive constructions and the German Perfekt with *sein* 'to be'.

We also adapted the AA to the feature set used in the TiGer DB[2] [Forst et al., 2004] (Dependency Bank) and a hand-crafted gold standard from the TüBa-D/Z[3] (TUBA100).

[2]The TiGer DB distinguishes 52 different grammatical features. We use a slightly modif ed version without the distinction between different prepositional objects, and without morphological features or compound analysis.

[3]The TüBa-D/Z gold standard was semi-automatically created by Heike Zinsmeister and Yannick Versley, using the conversion method of Versley [2005] on 100 randomly selected trees from the TüBa-D/Z. The feature set is similar to the TiGer DB.

4 LFG F-Structure Annotation and Evaluation on Two German Treebanks

For German, we adapted the AA to the node and edge labels of the two German treebanks. As described above, word order variation in German does not allow to make strong use of conf gurational information as in the English AA. Instead, we heavily rely on the grammatical function labels in the trees. This works well when annotating original treebank trees, but causes many problems when applied to parser output. State-of-the-art parsing results as presented in the PaGe Shared Task on Parsing German [Kübler, 2008] are in the range of 58-70% F-score for TiGer and 75-84% for TüBa-D/Z.[4] The differences in annotation schemes do not allow for a direct comparison of parsing results, but the message is clear: for both treebanks automatically assigned syntactic nodes and, even more important, grammatical function labels are to a great extent error-prone, which def nes an upper bound for treebank-based parsing into f-structures using the automatic annotation algorithm.

Section 4.2 presents parsing experiments with automatic LFG f-structure annotation based on TiGer and TüBa-D/Z, and evaluates the generated f-structures against hand-crafted gold standards from the TiGer treebank (TiGer DB, TIGER250) and from the TüBa-D/Z (TUBA100). However, before applying the AA to parser output we want to test its performance on gold standard syntax trees.

4.1 Results for LFG F-Structure Annotation on Gold Standard Syntax Trees

Table 3 shows results for automatic f-structure annotation on gold treebank trees for the sentences in the TiGer DB, the TIGER250 and the TUBA100.[5] Results for

	Prec.	Rec.	F-Score
TiGerDB	87.8	84.8	86.3
TIGER250	96.8	97.5	97.1
TUBA100	95.5	94.6	95.0

Table 3: Results for automatic f-structure annotation on gold treebank trees

the TIGER250 and the TUBA100 are quite good, while results for the TiGer DB are around 10% lower. This is due to mapping problems between the TiGer DB and TiGer treebank. The sentences in the TiGer DB have been converted semi-automatically into a dependency-based triple format, using a large, hand-crafted LFG grammar for German [Dipper, 2003] and then manually corrected. The TiGer DB provides a very f ne-grained description of linguistic phenomena in German,

[4] Results report constituent-based `evalb` labelled F-scores on syntactic nodes and grammatical function labels when using gold POS tags with gold GF labels as parser input

[5] We split the gold standards into development and test set, with 500 test set trees for the TiGer DB and 125 test trees for the TIGER250. Due to its limited size, we did not split the TUBA100.

but includes additional information which is not annotated in the TiGer treebank and thus cannot be derived automatically. This means that the TiGer DB-based evaluation is biased in favour of the hand-crafted LFG grammar of Dipper [2003].

4.2 Parsing German with Automatically Acquired LFG Grammars

In our experiments we use the Berkeley parser [Petrov and Klein, 2008], a language-agnostic parser which automatically ref nes and re-annotates the training data by applying split-and-merge operations, so that the likelihood of the transformed treebank is maximised. The Berkeley parser achieved the best results in the Shared Task on Parsing German (ACL 2008).

We removed the gold standard sentences from the treebanks and extracted two training sets with 25,000 sentences each. For TiGer we persued two different ways of resolving crossing branches in the trees: (1) by attaching the non-head child nodes higher up in the tree, following Kübler [2005], and (2) by splitting discontinuous nodes into smaller "partial nodes" [Boyd, 2007], a strategy which aims at preserving local tree structure while allowing the system to recover the original dependencies after parsing. With regard to GF labels we tested two different settings: in the f rst setting (Atomic) we merged categorial node labels with grammatical function labels and trained the parser on the new atomic labels. In the second setting (FunTag) we removed GF labels from the training data and trained the parser on syntactic categories only. The GF labels were then assigned in a post-processing step, using the SVM-based grammatical function labelling software by Chrupała et al. [2007]. We parsed the different test sets with the extracted grammars and, for the grammars without grammatical functions, let FunTag assign GFs to the parser output. The trees with grammatical function labels were passed over to the AA, where all nodes in the parse trees were annotated with LFG functional equations. Next we collected the equations and handed them over to a constrainst solver, which generated LFG f-structures.

4.2.1 Results

Table 4 shows constituent-based parsing results for the different test sets and settings (Atomic, FunTag) as well as results for f-structure evaluation. For the f rst setting, where we let the Berkeley parser assign the grammatical functions (Atomic), the two TiGer test sets yield constituent-based parsing results in the range of 76-79% (labelled F-score on syntactic categories) and 67-70% (including GF labels). Results for the TüBa-D/Z are more than 10% higher, which is an artifact of the different treebank annotation schemes and does not ref ect parser output quality, as can be seen in the f-structure evaluation. On the f-structure level precision is in the range of 73-81%, while recall for the TüBa-D/Z f-structures is dramatically lower at around 45%. For the TiGer, we achieve a recall of 73.7% for TiGer DB and of 79.7% for the TIGER250 test set.

Parsing results for the Berkeley parser trained on TiGer syntactic nodes only

Constituent-based evaluation						
	Atomic			**FunTag**		
length<= 40	**F-score**	**F-score** GF	**POS acc.**	**F-score**	**F-score** GF	**POS acc.**
TiGerDB	79.3	70.2	96.0	81.0	70.9	97.0
TIGER250	76.6	66.9	95.4	79.3	68.4	96.5
TUBA100	89.3	80.2	96.5	89.2	76.3	96.4
f-structure evaluation						
	Atomic			**FunTag**		
	Precision	**Recall**	**F-score**	**Precision**	**Recall**	**F-score**
TiBerDB	73.0	73.9	73.4	76.1	65.1	70.2
TIGER250	81.4	79.7	80.5	87.6	67.5	76.3
TUBA100	76.9	45.1	56.9	75.8	39.3	51.7

Table 4: C-structure parsing results (labelled F-score without and with GF) and f-structure evaluation

(FunTag) are higher than for the atomic labels. For TüBa-D/Z, however, we observe better results when training on both syntactic categories and grammatical functions. The FunTag-assigned GFs yield better `evalb` results and a higher precision for the TiGer f-structures. For the TüBa-D/Z, precision is slightly lower than for f-structures generated from parser output where the Berkeley parser did the function labelling. The better precision for the TiGer f-structures comes at the cost of a decrease in recall. For the TüBa-D/Z f-structures, recall is even lower than before.

There are several reasons for the low recall for the TüBa-D/Z: (1) Due to its limited size the TUBA100 does not cover all relevant grammatical phenomena and therefore is not suff cient as a test set for grammar development, which is ref ected in the low recall score. (2) Phrases without a clear dependency relation to the other constituents in the tree are attached directly to the root node in the TüBa-D/Z. The resulting tree structure makes it impossible for the AA to disambiguate the sentence and f nd a suitable dependency relation for the highly attached node, which means that these nodes are not represented in the f-structure, further lowering recall for the TüBa-D/Z. (3) NP internal structure in the TüBa-D/Z contains less information than in TiGer, where grammatical function labels distinguish genitive attributes, dative attributes and comparative complements. The missing information can be partly retrieved from morphological annotation, but this would require an extensive treebank transformation to make this information available to the parser. The grammars extracted from the treebanks do not include morphological information, which means that the TiGer grammars encodes more specif c functional information than the TüBa-D/Z grammars.

Yet another reason for the lower recall for TüBa-D/Z f-structures can be found in the design of the grammatical function labels used in the annotation. While the original treebanks use roughly the same number of grammatical functions (44 in TiGer versus 40 in TüBa-D/Z; Table 5), some of the grammatical functions in the TüBa-D/Z occur only with a very low frequency. When comparing two smaller subsets of 2,000 gold treebank trees, we still f nd 42 of the 44 GFs in

	Gold all	Gold 2000	Atomic	FunTag
TiGer	44	42	41	40
TüBa-D/Z	40	33	31	19

Table 5: Number of different grammatical functions in TiGer/TüBa-D/Z gold trees and reproduced in the different parsing settings (Atomic/FunTag)

the TiGer set, while the TüBa-D/Z subset uses only 33 of the 40 GFs. For parser output the problem gets even worse. In the TiGer-trained parser output for the same subset of 2,000 sentences we find 41 different GF labels when the Berkeley parser assigns the grammatical functions, and 40 when FunTag does the GF labelling, while in a data set of the same size from the TüBa-D/Z, only 31 different GF labels are used in the parser output (Atomic), and the FunTag approach yields only 19 different grammatical functions. This leads to a crucial difference between the type of information encoded in the GF labels for the two treebanks: while TiGer labels describe the grammatical function of one node, in TüBa-D/Z the GF labels (besides the main grammatical functions such as subject and acusative or dative object) express dependency relations between different nodes in the tree, which are often positioned in different topological fields. As pointed out, some of the grammatical functions in the TüBa-D/Z occur with a very low frequency.[6] This poses a problem for machine learning methods, which rely on a sufficiently large set of training instances in order to achieve good performance on unseen data.

GF	**Atomic**	**FunTag**	**Atomic**	**FunTag**
	TiGer (2,000 sent.)		*TüBa-D/Z (2,000 sent.)*	
DA	52.5	74.9	56.8	27.2
OA	79.5	85.5	69.0	46.4
SB	90.0	88.4	85.2	72.1
ALL GF	93.1	94.4	91.9	88.3

Table 6: Evaluation of main grammatical functions in TiGer and TüBa-D/Z (dative object: DA/OD, accusative object: OA, subject: SB/ON)

Next we compare results for the main grammatical functions (subject, accusative and dative object) on 2,000 sentence test sets from TiGer and TüBa-D/Z (Table 6). For parser-assigned GFs, we observe better results for dative objects (DA/OD) for the parsing model trained on the TüBa-D/Z, while for subjects and accusative objects the TiGer-trained parser yields better results. The SVM-based FunTag shows poor performance on the TüBa-D/Z data, while for TiGer the function labeller outperformes the setting where the Berkeley parser does the GF assignment (Atomic). This divergent behaviour might be due to the different data

[6] OA-MODK (conjunct of modifier of accusative object), ON-MODK (conjunct of modifier of nominative object) and OADVPK (conjunct of modifier of ADVP object) occur only once in 27,125 sentences in TüBa-D/Z Release 3, OG-MOD (modifier of genitive object) 7 times, OADJP-MO (modifier of ADJP object) 8 times, OADVP-MO (modifier of ADVP object) 10 times, and FOPPK (facultative object of PP object) 17 times.

structures in the treebanks. The split into topological f elds in the TüBa-D/Z takes away necessary context information, which is encoded in the feature set for the f at TiGer trees.

4.3 Different Approaches to Discontinuity and their Impact on F-Structure Annotation

Boyd [2007] presents an improved method for converting the crossing branches in TiGer into context-free representations by splitting up discontinuous nodes into marked "partial" nodes. She shows that the improved conversion results in more consistent trees and improves results in a labelled dependency evaluation for accusative, dative and prepositional objects. In her experiments, Boyd used an unlexicalised PCFG parsing model (LoPar, Schmid [2000]) with gold POS tags as parser input.

We applied the split-node conversion method to the TiGer data and trained the Berkeley parser on the converted training sets. Table 7 shows parsing results for the two conversion methods: (1) raised nodes and (2) split nodes. For the TiGer DB test set, results for the split-node conversion are slightly worse, while for the TIGER250 test set there is a small improvement of 1% F-score. For both data sets, however, the number of valid f-structures decreases considerably.

	Precision	**Recall**	**F-score**	**valid F-struc.**
		TiGer DB		
raised	73.0	73.9	73.4	82.4
split	71.8	72.0	71.9	71.0
		TIGER250		
raised	81.5	80.9	81.2	88.0
split	82.7	81.8	82.2	84.0

Table 7: f-structure evaluation on converted TiGer trees (raised- vs. split-node)

Boyd's split-node conversion works well for pure PCFG parsers like LoPar. The Berkeley parser, however, makes use of horizontal markovisation, which breaks up the original grammar rules and generates new rules which have not been seen in the training set. This also admits rules with only one of the two partial nodes, which means that a reconstruction of the original tree is impossible, and often leads to clashes during f-structure generation.

5 LFG Parsing: Related Work

This section discusses related work and shows how our research compares to the wide-coverage hand-crafted LFG grammar of Dipper [2003], Rohrer and Forst [2006], and Forst [2007] developed in the ParGram project [Butt et al., 2002]. The ParGram German LFG uses 274 LFG-style rules (with regular expression-based right-hand sides) and several lexicons with detailed subcategorisation information and a guessing mechanism for default lexical entries [Rohrer and Forst,

GF	ParGram up. bound	ParGram log. lin.	ParGram low. bound	TiGerDB	DCU250
da	67	63	55	44	38
gr	88	84	79	71	87
mo	70	63	62	65	73
oa	78	75	65	69	63
quant	70	68	67	67	78
rc	74	62	59	34	30
sb	76	73	68	74	79
preds only	79.4	75.7	72.6	72.7	78.6
	coverage on the NEGRA treebank (>20,000 sentences)				
	81.5	81.5	81.5	88.2	88.7

Table 8: F-scores for selected grammatical functions for the ParGram LFG (upper bounds, log-linear disambiguation model, lower bounds) and for two automatically acquired TiGer grammars

2006]. Preprocessing in the experiments reported in Rohrer and Forst [2006] includes modules for tokenisation, morphological analysis and manual marking of named entities, before the actual parsing takes place. An additional disambiguation component based on maximum entropy models is used for reranking the output of the parser. Forst [2007] tested parser quality on 1,497 sentences from the TiGer DB and reported a lower bound, where a parse tree is chosen randomly from the parse forest, an upper bound, using the parse tree with the highest F-score (evaluated against the gold standard), as well as results for parse selection done by the log-linear disambiguation model.

Table 8 shows results for the ParGram LFG and for the automatically induced grammars on selected grammatical relations and on all grammatical functions excluding morphological and other features (preds only). The automatically induced TiGer DB and DCU250-style grammars were trained on the full TiGer treebank (>48,000 sentences, excluding the test data). We report results for the test sets from the TiGer DB and the DCU250.

The hand-crafted LFG outperforms the automatically acquired grammars on most GFs for the TiGer DB, but results are not directly comparable. The TiGer DB-based evaluation is biased in favour of the hand-crafted LFG. Named entities in the ParGram LFG input are marked up manually, while for our grammars these multiword units often are not recognised correctly and so are punished during evaluation, even if part of the unit is annotated correctly. Furthermore, the hand-crafted ParGram LFG grammar was used in the creation of the TiGer DB gold standard in the f rst place, ensuring compatibility as regards tokenisation and overall linguistic analysis.

F-scores for the DCU250 are in roughly the same range as the ones for the hand-crafted grammar. For high-frequency dependencies like subjects (sb) or modif ers (mo), results of the two grammars are comparable. For low-frequency depen-

GF	ParGram up. bound	ParGram log. lin.	ParGram low. bound	TiGerDB	DCU250
da	67	63	55	58	50
gr	88	84	79	68	88
mo	70	63	62	63	77
oa	78	75	65	68	80
quant	70	68	67	58	69
rc	74	62	59	50	50
sb	76	73	68	76	85
preds only	79.4	75.7	72.6	76.0	84.4

Table 9: Precision for selected grammatical functions for the ParGram LFG and for the TiGer grammars

dencies like dative objects (da) or relative clauses (rc), however, the hand-crafted LFG outperforms the automatic LFG f-structure annotation algorithm by far. Coverage for the automatically acquired grammars is considerably higher than for the hand-crafted LFG grammar. Rohrer and Forst [2006] report a coverage of 81.5% (full parses) when parsing the NEGRA treebank, which contains newspaper text from the same newspaper as in the TiGer treebank. By contrast, the automatically acquired TiGer grammars achieve close to 90% coverage on the same data. On the TiGer treebank Rohrer and Forst [2006] report coverage of 86.4% full parses, raising the possibility that, as an effect of enhancing grammar coverage by systematically extracting development subsets from TiGer, the ParGram LFG is tailored closely to the TiGer treebank.

The DCU250 test set is equally biased towards the TiGer treebank-based LFG resources, as it only represents what is encoded (directly or implicitly) in the TiGer treebank. The truth is somewhere in between: The TiGer DB evaluation of the treebank-based LFG resources attempts to a limited extent to counter the bias of the original TiGer DB resource towards the hand-crafted LFG grammar by removing distinctions which cannot be learned from TiGer data only, and by relating TiGer DB to (some of) the original TiGer tokenisation using the version prepared by Boyd et al. [2007]. The resulting resource still favours the hand-crafted LFG resources, which outperform the treebank-based resources by about 3% points absolute. Looking at precision, results for the TiGer grammars are more or less in the same range as the F-scores for the Pargram LFG (Table 9).[7]

5.1 Discussion

Our automatically extracted grammars yield better coverage than the hand-crafted LFG of Dipper [2003], Rohrer and Forst [2006] and Forst [2007], but with regard to F-score the ParGram LFG still outperforms the automatically acquired gram-

[7]Unfortunately, Forst [2007] does not report results for precision and recall.

mars. The lower results for our grammars are not due to low precision: Table 9 contrasts F-scores for the Pargram LFG with results for precision as achieved by the automatically acquired TiGer grammars. Future work should therefore focus on improving recall in order to achieve results comparable with or better than hand-crafted grammars. One promising approach is the one of Seeker [2009], who describes a grammatical function labeller based on Integer Linear Programming (ILP). Seeker presents a two-step approach, consisting of a classif cation step and a selection step. During classif cation, the probability distribution over all possible labels for each node in the tree is computed, using a maximum entropy classif er. During selection, the overall probability of the whole tree is optimised, where the ILP-based approach allows the developer to implement hard constraints (e.g.: no more than one subject per local tree). First results show that global optimisation in combination with linguistically motivated constraints improves precision and coverage. F-scores for f-structure evaluation on the TiGer DB increase to more than 75%, while coverage was raised from around 88% to more than 96%.

An unsolved problem is the encoding of LDDs in treebank annotation schemes for (semi-) free word order languages. Currently, neither the TiGer treebank and even less so the TüBa-D/Z way of representing non-local dependencies can be learned successfully by statistical parsers. An approach to resolving LDDs at the f-structure level was described in Cahill et al. [2004] and Cahill [2004] and successfully implemented as part of the English treebank-based LFG acquisition and parsing architectures. However, the method of Cahill et al. relies on complete f-structures, which means that the recall problem must have been solved before we can reliably and prof tably compute LDDs on f-structure level for German.

6 Conclusions

We presented two architectures for the automatic acquisiton of LFG resources, based on two German treebanks. Compared to a hand-crafted German LFG, our method yields higher coverage and comparable results for the high-frequency grammatical functions, while for the less frequent GFs the hand-crafted grammar clearly outperforms the automatic approach.

We have outlined a number of problems for treebank-based f-structure annotation for German: (1) The semi-free word order in German rules out the use of conf gurational information for f-structure annotation. (2) Parsing results for German, especially for GF assignment, are not reliable enough to support accurate f-structure annotation. (3) Our alternative approach to assign GF labels using an SVM-based function labeller achieves high precision, but at the cost of recall. This is due to missing context sensitivity of the function labeller, resulting in the assignment of conf icting GFs.

We showed that particular treebank encoding schemes have a strong impact on the usability of the resources. We argue that the GF label set in the TüBa-D/Z, which has been designed with the aim of expressing dependency relations between

different nodes in the tree, is less adequate for the automatic acquisition of LFG resources than the label set in TiGer. The GF labels in the TüBa-D/Z are harder to learn and also encode less specif c grammatical information than the ones in TiGer.

The task of automatically inducing linguistic resources from (semi-) free word order languages is much harder than for more conf gurational languages like English. Future research needs to address the problem of automatic GF assignment which for German is far more important than for conf gurational languages (one promising line of research has been outlined in Section 5.1). Only then can we expect to automatically induce high-quality linguistic resources for languages other than English and other conf gurational languages.

References

Adriane Boyd. Discontinuity revisited: An improved conversion to context-free representations. In *Proceedings of the Linguistic Annotation Workshop (LAW 2007)*, pages 41–44, Prague, Czech Republic, 2007.

Adriane Boyd, Markus Dickinson, and Detmar Meurers. On representing dependency relations – insights from converting the German TiGerDB. In *Proceedings of the 6th International Workshop on Treebanks and Linguistic Theories (TLT-07)*, pages 31–42, Bergen, Norway, 2007.

Sabine Brants, Stefanie Dipper, Silvia Hansen, Wolfgang Lezius, and George Smith. The TIGER Treebank. In Erhard W. Hinrichs and Kiril Simov, editors, *Proceedings of the First Workshop on Treebanks and Linguistic Theories*, pages 24–42, Sozopol, Bulgaria, 2002.

Ted Briscoe and John Carroll. Robust accurate statistical annotation of general text. In *Proceedings of the 3rd International Conference on Language Resources and Evaluation (LREC-02)*, pages 1499–1504, Las Palmas, Canary Islands, 2002.

Michael Burke, Olivia Lam, Aoife Cahill, Rowena Chan, Ruth O'Donovan, Adams Bodomo, Josef van Genabith, and Andy Way. Treebank-based acquisition of a chinese lexical-functional grammar. In *Proceedings of the 18th Pacif c Asia Conference on Language, Information and Computation (PACLIC-18)*, pages 161–172, Tokyo, Japan, 2004.

Miriam Butt, Helge Dyvik, Tracy Holloway King, Hiroshi Masuichi, and Christian Rohrer. The parallel grammar project. In *Proceedings of COLING-02 Workshop on Grammar Engineering and Evaluation*, pages 1–7, Taipei, Taiwan, 2002.

Aoife Cahill. *Parsing with Automatically Acquired, Wide-Coverage, Robust, Probabilistic LFG Approximations*. PhD dissertation, School of Computing, Dublin City University, Dublin, Ireland, 2004.

Aoife Cahill, Mairéad McCarthy, Josef van Genabith, and Andy Way. Automatic annotation of the penn-treebank with LFG f-structure information. In *LREC-02 workshop on Linguistic Knowledge Acquisition and Representation - Bootstrapping Annotated Language Data, Third International Conference on Language Resources and Evaluation (LREC-02), post-conference workshop*, pages 8–15, Paris, France, 2002.

Aoife Cahill, Martin Forst, Mairéad McCarthy, Ruth O'Donovan, Christian Rohrer, Josef van Genabith, and Andy Way. Treebank-based multilingual unif cation-grammar development. In *Proceedings of the Workshop on Ideas and Strategies for Multilingual Grammar Development, at the 15th European Summer School in Logic Language and Information*, Vienna, Austria, 2003.

Aoife Cahill, Michael Burke, Ruth O'Donovan, Josef van Genabith, and Andy Way. Long-distance dependency resolution in automatically acquired wide-coverage PCFG-based LFG approximations. In *42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*, pages 319–326, Barcelona, Spain, 2004.

Aoife Cahill, Martin Forst, Michael Burke, Mairéad McCarthy, Ruth O'Donovan, Christian Rohrer, Josef van Genabith, and Andy Way. Treebank-based acquisition of multilingual unif cation grammar resources. *Journal of Research on Language and Computation; Special Issue on Shared Representations in Multilingual Grammar Engineering*, pages 247–279, 2005.

Aoife Cahill, Michael Burke, Ruth O'Donovan, Stefan Riezler, Josef van Genabith, and Andy Way. Wide-coverage deep statistical parsing using automatic dependency structure annotation. *Computational Linguistics*, 34(1):81–124, 2008.

Grzegorz Chrupała, Nicolas Stroppa, Josef van Genabith, and Georgiana Dinu. Better training for function labeling. In *Proceedings of the Conference on Recent Advances in Natural Language Processing (RANLP 2007)*, pages 133–138, Borovets, Bulgaria, 2007.

Stefanie Dipper. Implementing and documenting large-scale grammars — German LFG, doctoral dissertation, ims, university of stuttgart. *Arbeitspapiere des Instituts für Maschinelle Sprachverarbeitung (AIMS)*, 9(1), 2003.

Martin Forst. Filling statistics with linguistics - property design for the disambiguation of German LFG parses. In *Proceedings of the ACL Workshop on Deep Linguistic Processing*, pages 17–24, Prague, Czech Republic, 2007.

Martin Forst, Núria Bertomeu, Berthold Crysmann, Frederik Fouvry, Silvia Hansen-Schirra, and Valia Kordoni. Towards a dependency-based gold standard for German parsers - the TiGer Dependency Bank. In *Proceedings of the COLING Workshop on Linguistically Interpreted Corpora (LINC '04)*, pages 31–38, Geneva, Switzerland, 2004.

Julia Hockenmaier and Mark Steedman. Generative models for statistical parsing with combinatory categorial grammar. In *40th Annual Meeting of the Association for Computational Linguistics (ACL-02)*, pages 335–342, Philadelphia, PA, 2002.

Ronald M. Kaplan, Stefan Riezler, Tracy H. King, John. T. Maxwell III, Alexander Vasserman, and Richard Crouch. Speed and accuracy in shallow and deep stochastic parsing. In *Proceedings of the Human Language Technology Conference and the 4th Annual Meeting of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL-04)*, pages 97–104, Boston, MA, 2004.

Sandra Kübler. How do treebank annotation schemes inf uence parsing results? Or how not to compare apples and oranges. In *Proceedings of the 5th International Conference on Recent Advances in Natural Language Processing (RANLP 2005)*, pages 293–300, Borovets, Bulgaria, 2005.

Sandra Kübler. The PaGe 2008 shared task on parsing German. In *ACL Workshop on Parsing German (PaGe-08)*, pages 55–63, Columbus, OH, 2008.

David M. Magerman. Statistical decision-tree models for parsing. In *33rd Annual Meeting of the Association for Computational Linguistics (ACL-95)*, pages 276–283, Cambridge, MA, 1995.

Hiroko Nakanishi, Yusuke Miyao, and Jun'ichi Tsujii. Using inverse lexical rules to acquire a wide-coverage lexicalized grammar. In *IJCNLP 2004 Workshop on Beyond Shallow Analyses - Formalisms and Statistical Modeling for Deep Analyses*, Sanya City, Hainan Island, China, 2004.

Ruth O'Donovan, Michael Burke, Aoife Cahill, Josef van Genabith, and Andy Way. Large-scale induction and evaluation of lexical resources from the penn-II and penn-III treebanks. *Computational Linguistics*, 31(3):329–366, 2005.

Slav Petrov and Dan Klein. Parsing German with language agnostic latent variable grammars. In *ACL Workshop on Parsing German (PaGe-08)*, pages 33–39, Columbus, OH, 2008.

Christian Rohrer and Martin Forst. Improving coverage and parsing quality of a large-scale LFG for German. In *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC-06)*, pages 2206–2211, Genoa, Italy, 2006.

Anne Schiller, Simone Teufel, and Christine Thielen. Guidelines für das tagging deutscher textkorpora mit STTS. Technical report, Universität Stuttgart and Universität Tübingen, Tübingen, Germany, 1995.

Helmut Schmid. LoPar: Design and implementation. Technical report, Universität Stuttgart, Stuttgart, Germany, 2000.

Wolfgang Seeker. *On the Use of Hard Linguistic Constraints in Automatic Grammar Acquisition for German*. Diploma thesis, Institut für Linguistik, Potsdam, Germany, 2009.

Heike Telljohann, Erhard W. Hinrichs, Sandra Kübler, and Heike Zinsmeister. *Stylebook for the Tübingen Treebank of Written German (TüBa-D/Z)*. Universität Tübingen, Germany, 2005.

Yannick Versley. Parser evaluation across text types. In *Proceedings of the 4th Workshop on Treebanks and Linguistic Theories (TLT-05)*, pages 209–220, Barcelona, Spain, 2005.